QuickTime Streaming Server 4

Session 812

QuickTime Streaming Server 4

Chris LeCroy
Engineering Manager,
Streaming Servers

Introduction

- Brief review of new features in QTSS-4
- Technical information

Technical Things You Will Learn

- Server modules overview
- Admin protocol overview

What Is New in QTSS-4

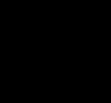
- Module API enhancements and more developer documentation
- Skip protection enhancements
- Performance improvements
- Completely redesigned Web Admin
- Automated relays
- Native MP3 streaming
- MPEG-4 support

Module API Enhancements and More Documentation

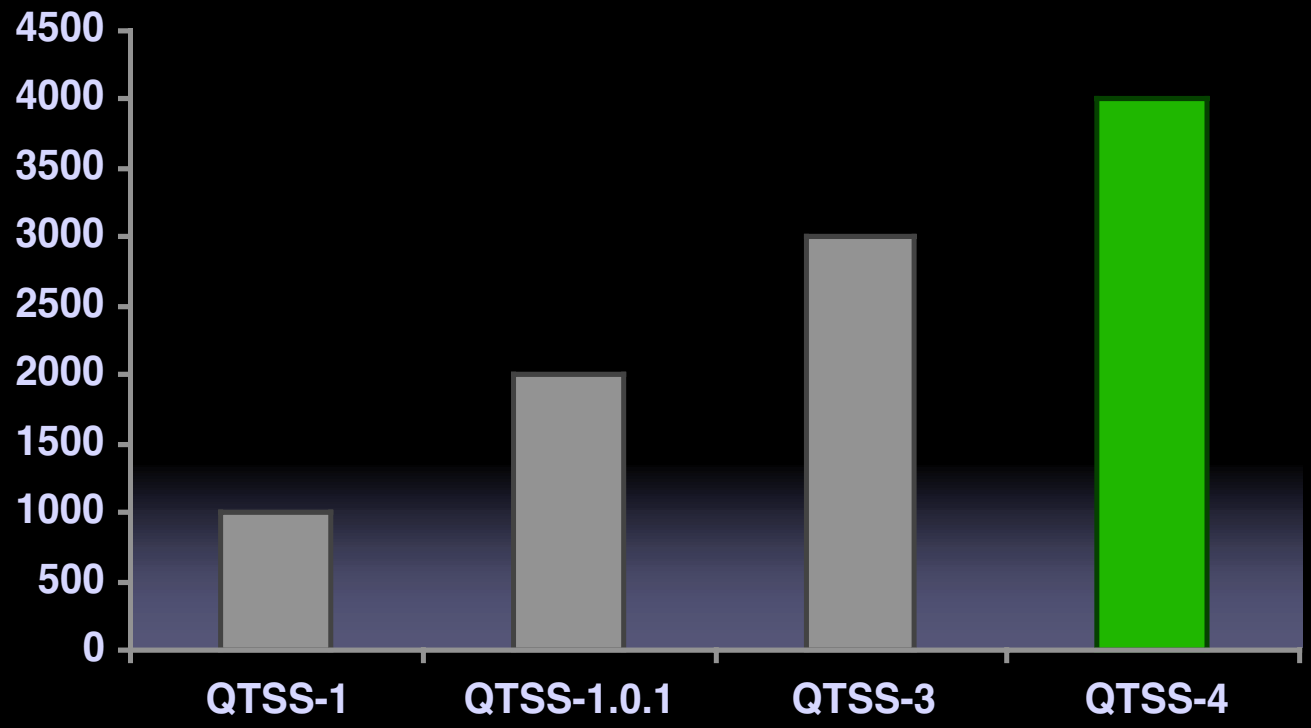
- Easier to build more powerful modules
  - Modules can now create their own QTSS objects
- New Documentation
  - Reliable RTP Protocol (Skip Protection)
  - Tunneling of RTSP via HTTP
  - Caching protocol extensions

Performance Improvements

- Up to 4,000 concurrent streams from a single Mac OS X Server
- QTSS-4 doubles the number of high bit-rate movies that can be streamed from a single server
- “Optimize for Server” is only necessary for servers that will be under extremely heavy load

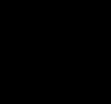


Performance Improvements



Redesigned Administration App

- A new look
- Easy Setup Assistant
- Relay setup and monitoring
- Completely revamped Play List management



MP3 Streaming

- Native MP3 streaming—
  Icecast/Shoutcast compatible
- Easy to set up internet radio stations
- Works with all the major MP3 players
  (iTunes, WinAmp, WMP, RealOne, etc.)
- Allows insertion of ads or station IDs
- Real-time play list manipulation
- Broadcast locally or to a remote server

Automated Relays

- Easy to setup in web admin
- Automatic negotiation between relays
- Pull external feeds into your intranet
- Send to multiple destinations—
  unicast and multicast

QTSS Admin Protocol

John Anderson
QuickTime Streaming Server Engineering

QTSS Admin Protocol

- Allows other applications to interact with QTSS
- Uses simple HTTP requests
- Incorporates preferences and data from installed modules

Demo

John Anderson
QuickTime Streaming Server Engineering

QTSS Inspector

Cocoa application for browsing the Admin Protocol values

- Based on the NSOutlineView class
- Gets its data using an HTTP request

QTSS Admin Protocol Format

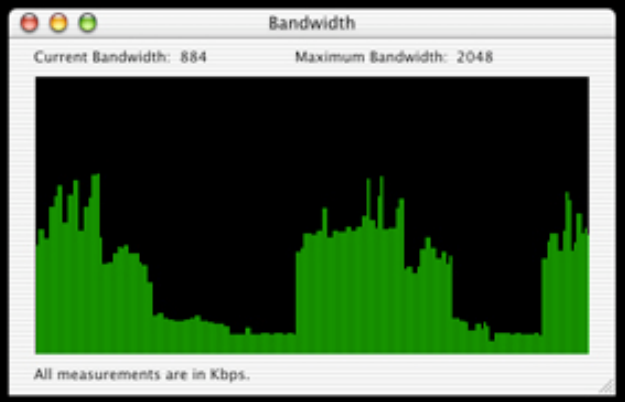
- Accessible through any port on which QTSS is serving (typically 554)
- All URL paths begin with /modules/admin
- Can use “wildcards” to get multiple values or sets
- Sample URL:
  http://127.0.0.1:554/modules/admin/server/*

Demo

John Anderson
QuickTime Streaming Server Engineering

Cocoa Monitoring App

Cocoa application for graphically monitoring the server's current bandwidth



- Fetches the current bandwidth every 1/2 second
- Custom NSView subclass displays a historical chart

Cocoa Monitoring App

Composition

- NSView subclass constructs a bar graph from history array
- NSTimer updates the history array and tells the NSView to redraw itself
- AdminProtocolAccessObj instance remembers login information and handles HTTP requests

Parameters in Server

- URL for current bandwidth cap:
  http://host:554/modules/admin/server/qtssSvrPreferences/maximum_bandwidth
- URL for RTSP bandwidth:
  http://host:554/modules/admin/server/qtssRTPSvrCurBandwidth
- URL for MP3 bandwidth:
  http://host:554/modules/admin/server/qtssMP3SvrCurBandwidth

Getting Data

- In this example, HTTP connections are handled by a pre-made protocol object

```
// get the maximum connections
// will parse the HTTP result into an array and put it in "result"

[adminProtocolAccessObj getValueUsingHTTP:
        @"/modules/admin/server/qtssSvrPreferences/maximum_bandwidth"
                                withResult:result];

// set the field at the top of the window

[myMaxField setStringValue:[result objectAtIndex:0]];
```

Writing QTSS Modules

Mythili Devineni
QuickTime Streaming Server Engineering

Why Write Modules?

- All features are implemented in modules
- Add your much-needed feature not already present
- Sell your dynamic module
- Sell services based on QTSS with your added features

What Is a QTSS Role?

- Modules perform some processing in a QTSS Role
- It provides a well-defined state
- The server passes on one or more objects to the module in each role
- Implement features by registering for roles

A Few Ideas . . .

- Load balancing

  QTSS_RTSPFilter_Role

- Multiple content directories

  QTSS_RTSPRoute_Role

- Advanced Logging

  QTSS_ClientSession_Closing_Role

What Is a QTSS Object?

- Provides a way to exchange information between server and modules
- Consists of attributes that store data
- Built-in objects defined by server

  QTSS_RTSPRequestObject

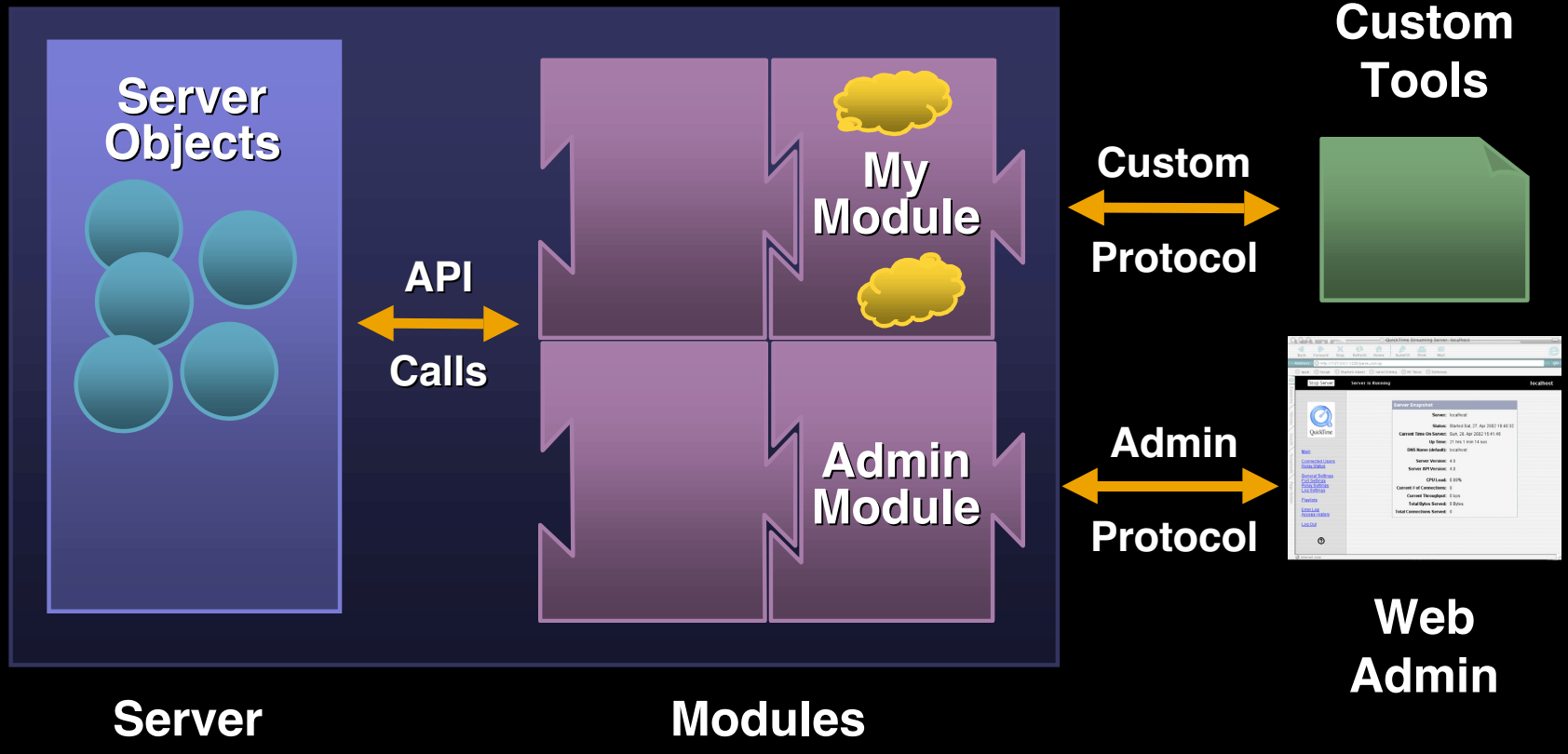
  QTSS_ServerObject

  QTSS_ClientSessionObject, etc.

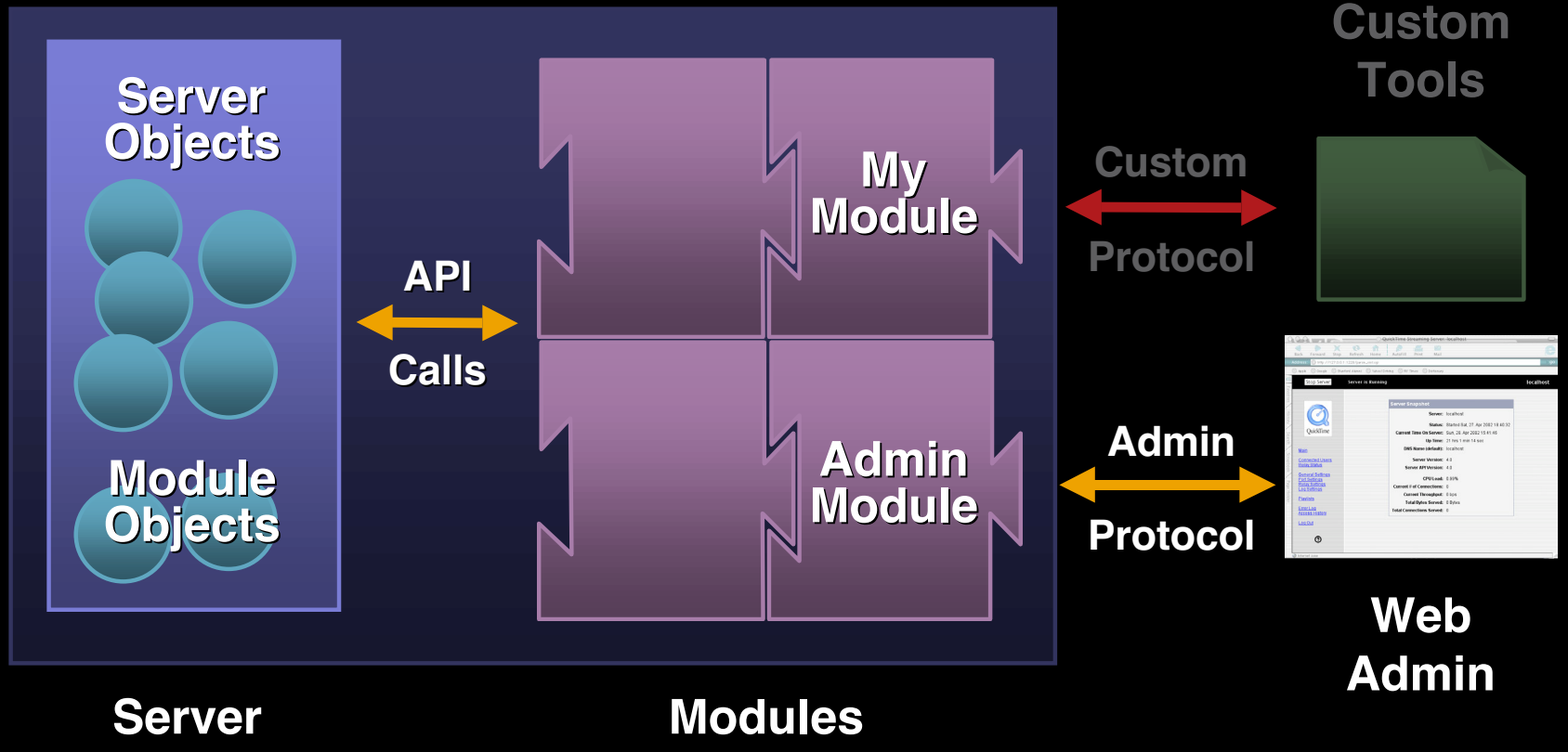
Why Create QTSS Objects?

- Store your data in QTSS objects
- Access the full capabilities of QTSS objects
- Objects automatically accessible through the Admin Protocol

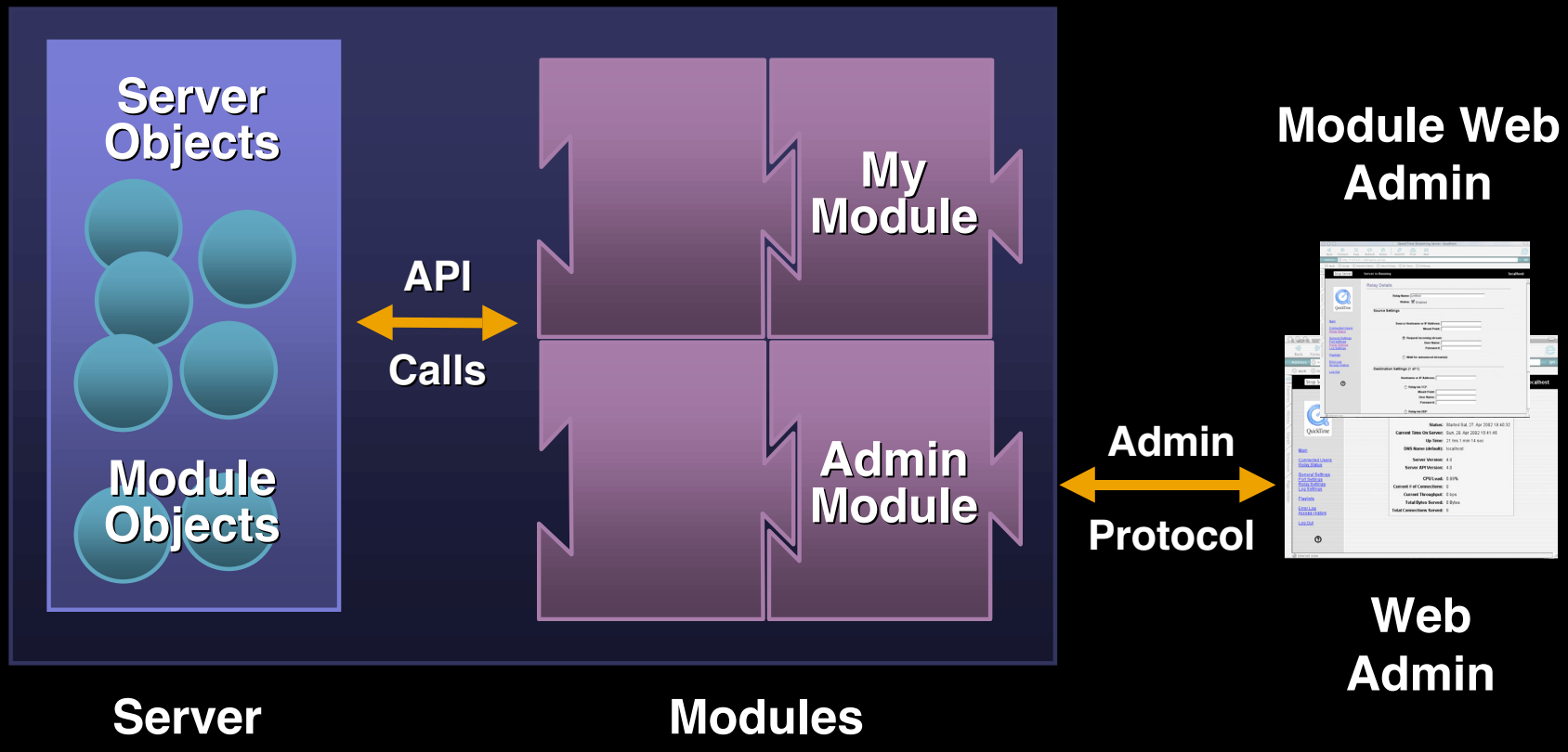
Before Module Objects . . .



With Module Objects . . .



With Module Objects—2



Creation of QTSS Objects API

- Define new object type or “Class”

  QTSS_CreateObjectType

- Instantiate an object of the defined type

  QTSS_CreateObjectValue

- Use any QTSS API call on your objects

Sample Module

QTSSHomeDirectoryModule

- Serve content from users' home directories (**i.e., rtsp://server.com/~joe/bday.mov**)
- Role
  QTSS_RTSPRoute_Role
- QTSS_RTSPRequestObject attributes
  qtssRTSPReqFilePath
  qtssRTSPReqRootDir

Sample Module Code

QTSSHomeDirectoryModule

- Register for the Role

```
QTSS_Error Register(QTSS_Register_Params* inParams)
{
    // Do role & attribute setup
    (void)QTSS_AddRole(QTSS_Initialize_Role);
    (void)QTSS_AddRole(QTSS_RTSPRoute_Role);
    …
}
```

Sample Module Code—2

QTSSHomeDirectoryModule

- Dispatch function for the Role

```
QTSS_Error QTSSHomeDirectoryDispatch
    (QTSS_Role inRole, QTSS_RoleParamPtr inParams)
{
   switch (inRole)
   { …
     case QTSS_RTSPRoute_Role:
          return RewriteRequestFilePathAndRootDir(
                              &inParams->rtspRouteParams);
     …
```

Sample Module Code—3

QTSSHomeDirectoryModule

- Set the request root directory

```
RewriteRequestFilePathAndRootDir(
                                &inParams->rtspRouteParams) {
  if (theRequestPathParser.PeekFast() == '~') {
    theRequestPathParser.Expect('~');
    theRequestPathParser.ConsumeUntil(&theFirstPath, '/');

    StrPtrLen theHomeDir;
    if (FindHomeDir(&theFirstPath, &theHomeDir)) {
        RewriteRootDir(inParams, &theHomeDir);
    …
```

Demo

Mythili Devineni
QuickTime Streaming Server Engineering

Roadmap

607 QuickTime and MPEG-4: A Technical Overview	Room A2 **Fri., 3:30pm**
FF010 QuickTime	Room J1 **Fri., 10:30am**

Who to Contact

Jeff Lowe
Quicktime Technology Manager
jefflowe@apple.com

http://developer.apple.com/wwdc2002/urls.html

For More Information

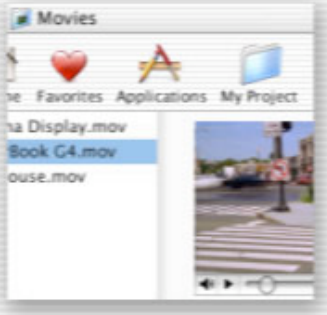
- Apple Open Source
  http://www.opensource.apple.com
- QuickTime Developers
  http://developer.apple.com/quicktime
- Mailing lists
  http://lists.apple.com
  - **Streaming-Server-Developers**
  - **Streaming-Server-Users**
  - **QuickTime-Talk**

Technical Documentation

- QuickTime Streaming Server API
- “Documentation” directory in source code
- Sample modules included with source code

Q&A





Jeff Lowe
QuickTime Technology Evangelist
jefflowe@apple.com

http://developer.apple.com/wwdc2002/urls.html


WWDC2002


WWDC2002


WWDC2002